



Authentication with Axis2 ADB

Version	1.0, 18 August 2007
Language	Java
Platform	Platform Independent
Tested On	Ubuntu, Tomcat 5.5, Axis2 1.3, Java 1.6
Author	Craig Mason-Jones (craig@lateral.co.za)
Licence	http://creativecommons.org/licenses/by-nc-sa/2.0/za/

Summary

Apache Axis2 provides a very easy route to using SOAP services from Java, but I've not found any useful information on the web about handling BASIC authentication when using them, and nothing about how to do it using the latest version of Axis2, version 1.3.

Solution

When I was trying to achieve basic authentication with Axis2, I found a few hints on the web related to using Authentication with Axis, but most of these seemed outdated, and in any case, I wanted to use the ADB generated classes, and I couldn't find any information on this.

Setting up a SOAP service requiring authentication

I'm calling the basic Axis2 Version service, that I've installed on my localhost tomcat. I've added BASIC authentication to the local Tomcat installation by adding the following snippet to my webapps/axis2/WEB-INF/web.xml file:

```

1  <web-app>
2
3     <!-- other elements -->
4
5     <security-constraint>
6     <web-resource-collection>
7         <url-pattern>/services/*</url-pattern>
8     </web-resource-collection>
9     <auth-constraint>
10    <role-name>soap</role-name>
11    </auth-constraint>
12    </security-constraint>
13
14    <login-config>
15    <auth-method>BASIC</auth-method>
16    <realm-name>Axis2 Services</realm-name>
17    </login-config>
18
19
20 </web-app>
21

```

I'm using the existing Tomcat users, so I have to create a role of 'soap' and add an appropriate user in the conf/tomcat-users.xml file. I add these lines:

```

22 <tomcat-users>
23 <!-- other elements -->
24 <role rolename="soap"/>
25 <user username="soap" password="land" roles="soap"/>
26 </tomcat-users>
27

```

Now I restart Tomcat, and use a browser to access <http://localhost:8180/axis2/services/Version?wsdl>. If everything has worked, I should be required to authenticate with the username 'soap' and password 'land'.

Creating the ADB proxy class from the WSDL file

To create the ADB proxy class from the WSDL file, I save the WSDL file to my system as `Version.wsdl`. I ensure that my `AXIS2_HOME` and `JAVA_HOME` environment variables are correctly set, then I generate the client proxy with:

```
28 $AXIS2_HOME/bin/wsd12java.sh -uri Version.wsdl -p auth -d adb -S src
```

These parameters are

Parameter	Description
<code>-uri Version.wsdl</code>	The wsdl file to use to generate the client proxy class.
<code>-p auth</code>	Place the generated files in the auth package.
<code>-d adb</code>	Generate ADB classes.
<code>-S src</code>	Place the generate source files off the src directory.

This generates three java files in `src/auth`.

Using authentication with the generated proxy classes

To use the SOAP service, I use the following `Main.java`:

```
29 package example;
30
31 import auth.*;
32
33 public class Main {
34     public static void main(String args[]) {
35         try {
36
37             VersionStub stub = new VersionStub();
38
39             org.apache.axis2.transport.http.HttpTransportProperties.Authenticator auth =
40 new org.apache.axis2.transport.http.HttpTransportProperties.Authenticator();
41 auth.setUsername("soap");
42 auth.setPassword("land");
43 auth.setPreemptiveAuthentication(true);
44 stub._getServiceClient().getOptions().setProperty(
45     org.apache.axis2.transport.http.HTTPConstants.AUTHENTICATE,
46     auth);
47
48             VersionStub.GetVersionResponse response = stub.getVersion();
49             System.out.println(response.get_return());
50         } catch (Exception X) {
51             System.err.println(X.getMessage());
52         }
53     }
54 }
```

The authentication code is in bold: lines 39 through 46. The `setPreemptiveAuthentication(true)` does not appear to be necessary, but it is, I suppose, good form.

Conclusion

Not very difficult, when you know how. The `org.apache.axis2.transport.http.HttpTransportProperties.Authenticator` class also supports DIGEST and NTLM authentication, but I've not played with these. Perhaps someone else would like to contribute to this article and add some notes on how these work...?